REMARKS ON EFFICIENT NUMERICAL ALGORITHMS FOR ULTRACOMPUTERS

by

Charles S. Peskin and Olof B. Widlund
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, N.Y. 10012

1. Introduction. Our purpose is to give an outline of a few numerical algorithms for some important problems arising in continuum mechanics which can take full advantage of the special features of ultra computers. We will discuss fast Poisson solvers, capacitance matrix or imbedding methods for special elliptic problems on general bounded domains and a finite difference approximation to the time dependent three dimensional Navier Stokes equations. We will show that algorithms exist which are within a logarithmic factor of achieving optimal speedup and which are relatively easy to implement. We have previously contributed to the development of similar algorithms and programs for serial computers and used these programs regularly to solve interesting applied problems in computational physics. We conclude with a few general comments on the parallel solution of linear systems of algebraic equations.

We will assume throughout that one processor is available for each element of our mesh functions, or arrays, to be processed. If this is not the case the speedup will clearly be limited in direct proportion to the ratio of the array dimension and the number of processors.

2. Fast Poisson Solvers. Since the mid sixties very fast methods have been developed for the highly structured linear systems of equations which arise when Laplace's equation, the biharmonic equation and certain other special elliptic problems are discretized by finite difference or finite element methods on rectangular regions covered by regular meshes. Storage required is at a minimum and the execution times are comparable to those of a few steps of some very simple iterative method. These codes are used extensively in many applications such as fluid dynamics and plasma physics. These methods are, however, limited to elliptic operators and regions such that the variables can be separated.

Among the basic building blocks for fast Poisson solvers are the Fast Fourier Transform (FFT) and the odd-even reduction algorithm for the solution of tridiagonal systems of equations. In the case of the three dimensional Poisson equation in Cartesian coordinates, we can consecutively apply the FFT to the mesh function of data in the three coordinate directions of the corresponding three dimensional array. A natural way of reordering the array between the FFT steps amounts to a cyclic permutation of the three coordinate axes. If the indexing of the three dimensional array is done as in FORTRAN and the indices are given in binary form these special permutations can be carried out by repeated use of the fundamental shuffle permutation. As shown in Schwartz's "ultracomputers", ultra computers lend themselves very well to the use of the FFT algorithm which can be easily imple-

mented in a logarithmic number of machine cycles. To complete the solution of Poisson's equation, element-wise multiplication by constants and an inverse three dimensional FFT are applied.

The FFT algorithm might not be applicable to all the variables if other coordinate systems are used. If all but one of the coordinates can be separated by FFT, a number of tridiagonal linear systems of equations will typically remain to be solved. Several algorithms which are of a comparable speed to that of the FFT are available for such systems; see Heller, SIAM Review, V. 20, 1978, pp. 740-777. The odd-even reduction methods appear particularly well suited since their structure has much in common with the FFT algorithm.

3. Capacitance Matrix or Imbedding Methods. These methods have been developed to extend the usefulness of fast Poisson solvers for problems on general regions. For a detailed account, we refer to O'Leary and Widlund, Math. Comp., 1979, to appear, (also available as a DOE-NYU report, COO-3077-155). Extensive theoretical and experimental evidence shows that iterative versions of this algorithm converge in a moderate number of steps which grows only in proportion to the logarithm of the number of degrees of freedom.

In each step a fast Poisson solver is used once or twice on a simple region in which the region of interest is imbedded. In addition a few sparse matrix operations are carried

out. Some of these can be described as the creation of a three dimensional array which is zero except at mesh points close to the boundary. At these exceptional points, values are obtained by using a few arithmetic operations on data obtained from neighboring mesh points. These data can easily be accessed using the shuffle connection and the fundamental shuffle permutation in a way quite similar to that outlined in the previous section. The other type of sparse matrix operations corresponds to the application of finite difference operators to a given array and are thus quite similar to those just considered.

In addition any finite difference or finite element method requires the generation of the discrete problem from information on the geometry of the region etc. In our experience, with serial computers, considerable care should be taken with this part of the algorithm to make it efficient and reliable. It appears, however, as if these questions can be solved satisfactorily for an ultra computer if the region is defined in terms of inequalities or the characteristic function of the region. We also note that capacitance matrix methods currently are actively being developed for several other important special elliptic problems. These methods are also very useful to accelerate the convergence of the iterative solution of rather general elliptic problems.

4. Incompressible Fluid Dynamics. In this section, we will consider the implementation of Chorin's finite difference

method for the time dependent Navier-Stokes equations in three space dimensions, see Chorin, Math. Comp., Vol. 22, 1968, pp. 745-762. We note that we can easily envision similar equally efficient implementations for other difference approximations to the same and similar problems.

The domain occupied by the fluid will be a unit cube, with periodic boundary conditions. This problem might appear too simple to be of interest in applications, but realistic problems can be treated by imbedding methods similar to those of section 3. In the past, we have had very good experience with this approach; see Peskin, Comp. Phys., Vol 25, 1977, p. 220-252. This paper describes a numerical method for the flow of blood in the heart. A realistic moving geometry is imbedded in a periodic domain, and a subroutine for the Navier-Stokes equations on this domain is called at each time step.

Chorin's finite difference method for the Navier-Stokes equations may be written as follows:

$$(1)\quad [I + \Delta t(u_x^n D_{ox} - \frac{1}{R} D_{+x} D_{-x})]\underline{u}^* = \underline{u}^n$$

$$(2)\quad [I + \Delta t(u_y^n D_{oy} - \frac{1}{R} D_{+y} D_{-y})]\underline{u}^{**} = \underline{u}^*$$

$$(3)\quad [I + \Delta t(u_z^n D_{oz} - \frac{1}{R} D_{+z} D_{-z})]\underline{u}^{***} = \underline{u}^{**}$$

$$(4)\quad \begin{aligned} \underline{u}^{n+1} &= \underline{u}^{***} - \Delta t \underline{G} p^{n+1} \\ D\underline{u}^{n+1} &= 0 \end{aligned}$$

where D_o, D_+, D_- are the centered, forward, and backward divided difference operators, and where D and $\underline{G}$ correspond-

ing to divergence and gradient are given by

$$(5) \qquad D\underline{u} = D_{ox}u_x + D_{oy}u_y + D_{oz}u_z$$

$$(6) \qquad \underline{G}p = (D_{ox}p,\ D_{oy}p,\ D_{oz}p) \ .$$

Implementation of Chorin's method involves the solution of the linear systems (1), (2), (3) and (4). Let $N = 2^p$ be the number of mesh points in each space direction and suppose that each mesh point corresponds to one processor of an ultracomputer. Then the system (1) consists of N^2 uncoupled tridiagonal systems of order N, one for each row of the mesh parallel to the x-axis. These systems can be solved by the odd-even reduction method, as indicated in section 2, in a time proportional to log N . The systems (2) and (3) are solved similarly after the permutation of the coordinate axes, see section 2, at the expense of log N fundamental shuffle permutations.

To explain how the system (4) is solved, we first eliminate $\underline{u}^{n+1}$ to obtain

$$(7) \qquad D\underline{G}p^{n+1} = \frac{1}{\Delta t} D\underline{u}^{***}$$

which is a discretization of Poisson's equation. Formation of the right hand side involves differencing the function $\underline{u}^{***}$ in all 3 space directions. In the ultracomputer, differences in the x direction will be readily available, and differences in the y and z directions can be obtained after permuting the axes. An excellent way to solve for p is

to use a fast Poisson solver; see section 2 . Finally, we apply $\underline{G}$ to p (again, permuting the axes) and evaluate $\underline{u}^{n+1}$. The total machine cycle count for one time step is thus proportional to log N .

It should be mentioned that this speed could not be achieved on a parallel computer with a built-in three dimensional architecture with connections only between neighboring processors. In that case the tridiagonal systems should be solved by a conventional tridiagonal Gaussian elimination method and on the order of N machine cycles would be required. Similar problems of communication would affect the FFT algorithm and the accurate solution of the Poisson equation by any other method.

5. Some Remarks on Linear Algebraic Systems of Equations. A frequently occuring problem in scientific computation is the solution of general dense linear systems of equations. It is easy to show that Gaussian elimination of a system of n variables can be carried out in n steps of parallel computation. The authors know of no alternative method which substantially decreases this machine cycle count while promising to be numerically stable. Further important progress on this problem would probably greatly influence the choice of algorithms for the solution of linear systems arising in continuum mechanics.

Linear systems with band structure which require no pivoting can be solved by methods analogous to the odd-even

reduction method for tridiagonal matrices. By using a linear method for dense submatrices this can easily be accomplished in a time proportional to the product of the bandwidth and a logarithmic factor. If the graph of the coefficient matrix is planar or of finite element type, nested dissection algorithms can be used to make the solution time proportional to the square root of the number of unknowns; see George, SIAM J. Numer. Anal., Vol. 10, 1973, pp. 345-367 and Lipton and Tarjan, SIAM J. Appl. Math., Vol. 36, 1979, pp. 177-189. It is interesting to note that in this ultracomputer context that no order of magnitude advantage seems to be offered by using nested dissection rather than more conventional band schemes.

Many classical iterative methods can be implemented successfully on ultracomputers as well as on other parallel computers. Important building blocks are sparse matrix operations similar to those discussed in section 3. Since a number of iterative methods require the solution of sparse triangular systems, the use of the interesting algorithms for such systems which are discussed in Heller's paper, see reference in section 2, should be explored in the ultracomputer context. We also note that a study of block conjugate gradient type methods and other block iterative algorithms would be of interest in particular if the number of processors exceed the order of the linear systems or if data between pairs of processors can be transferred in parallel.